Stacking context in CSS

What happens in the z-axis?

I’m Maria Paz.

UX Engineer

Or Front-end developer as a shorthand.

Women in Tech + Home-made Barista + Long-life learner

Linkedin // Instagram
@mariapazmp

Stacking Context

How elements are rendered in the screen.

Let’s Code!
codepen.io/mariapazz

Rendering on screen

Typically elements are rendered vertically in the same order we define them in HTML (document flow).

CSS position values like *static* or *relative* don’t “broke” the usual flow of the elements.

The position on screen also depends on the position of the parent element.

Elements are stacked in the y axis.

Rendering on screen

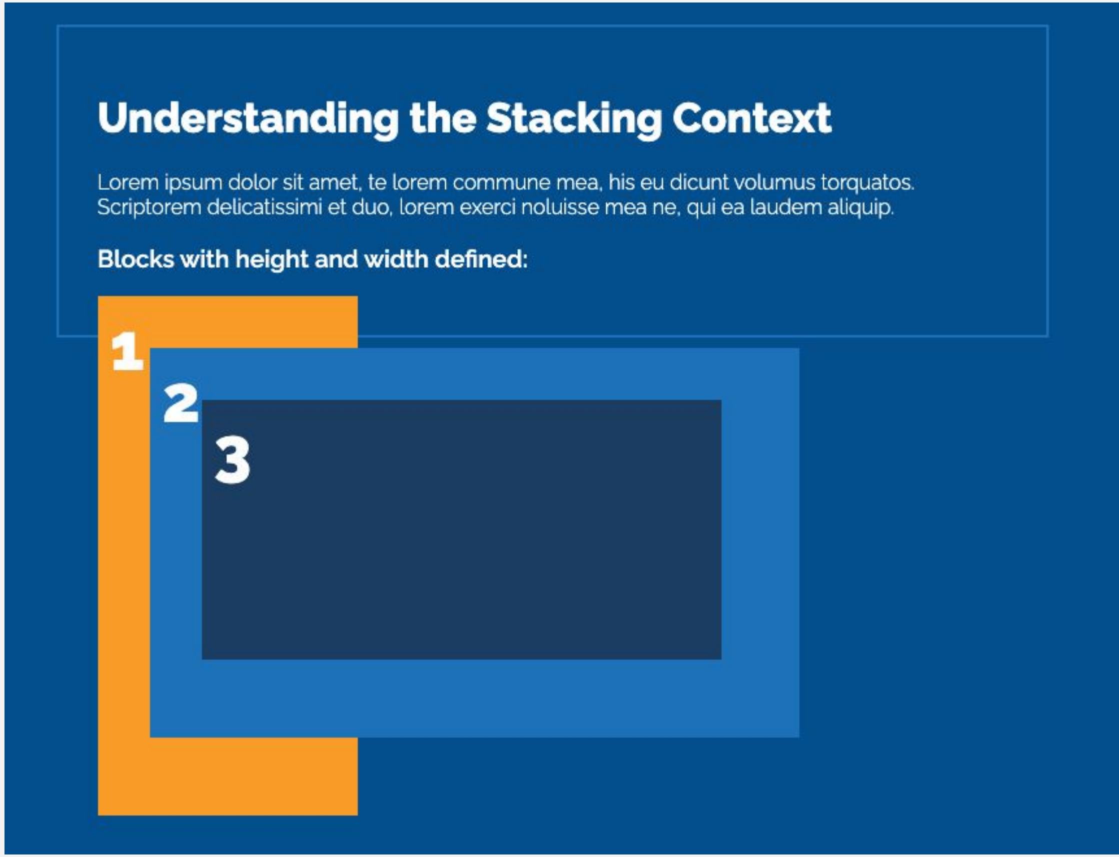
CSS position values like *absolute, fixed or sticky* can cause a different behavior. In this case, the position of the element depends of the parent's position, not the sibling's position.

And sometimes, elements can appear one on top of the other.

A new stacking context is created

With position: absolute elements appear one on top of the other.

It means, a new way to “stack” things is created.


Understanding the Stacking Context

Lorem ipsum dolor sit amet, te lorem commune mea, his eu dicunt volumus torquatos. Scriptorem delicatissimi et duo, lorem exerci noluisse mea ne, qui ea laudem aliquip.

Blocks with height and width defined:

When:
Stacking Context

When a stacking context is created?

A new stacking context is created when:

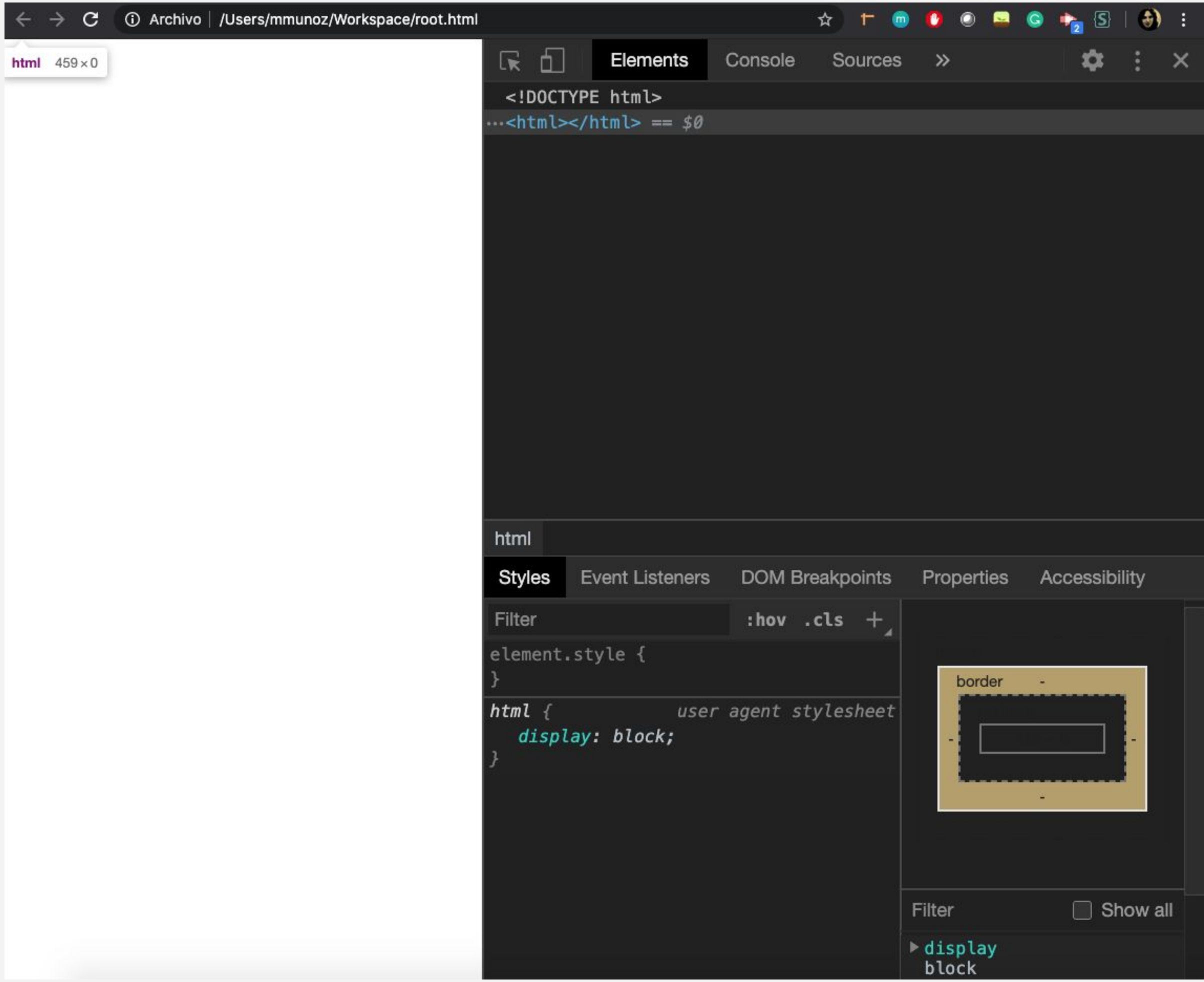
Full list at:
https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Positioning/Understanding_z_index/The_stacking_context

- Defining the root element (HTML)
- opacity < 1
- position: fixed
  position: sticky
- $$\left.\begin{array}{l}\text{position: absolute}\\ \text{position: relative}\end{array}\right\}\ \text{z-index != auto}$$

Stacking context when:

The root element of the document is defined.

```
<html></html>
```



Archivo | /Users/mmunoz/Workspace/root.html

html 459 × 0

Elements Console Sources

```
<!DOCTYPE html>
…<html></html> == $0
```

html

Styles Event Listeners DOM Breakpoints Properties Accessibility

Filter :hov .cls +

```
element.style {
}
html {                    user agent stylesheet
   display: block;
}
```

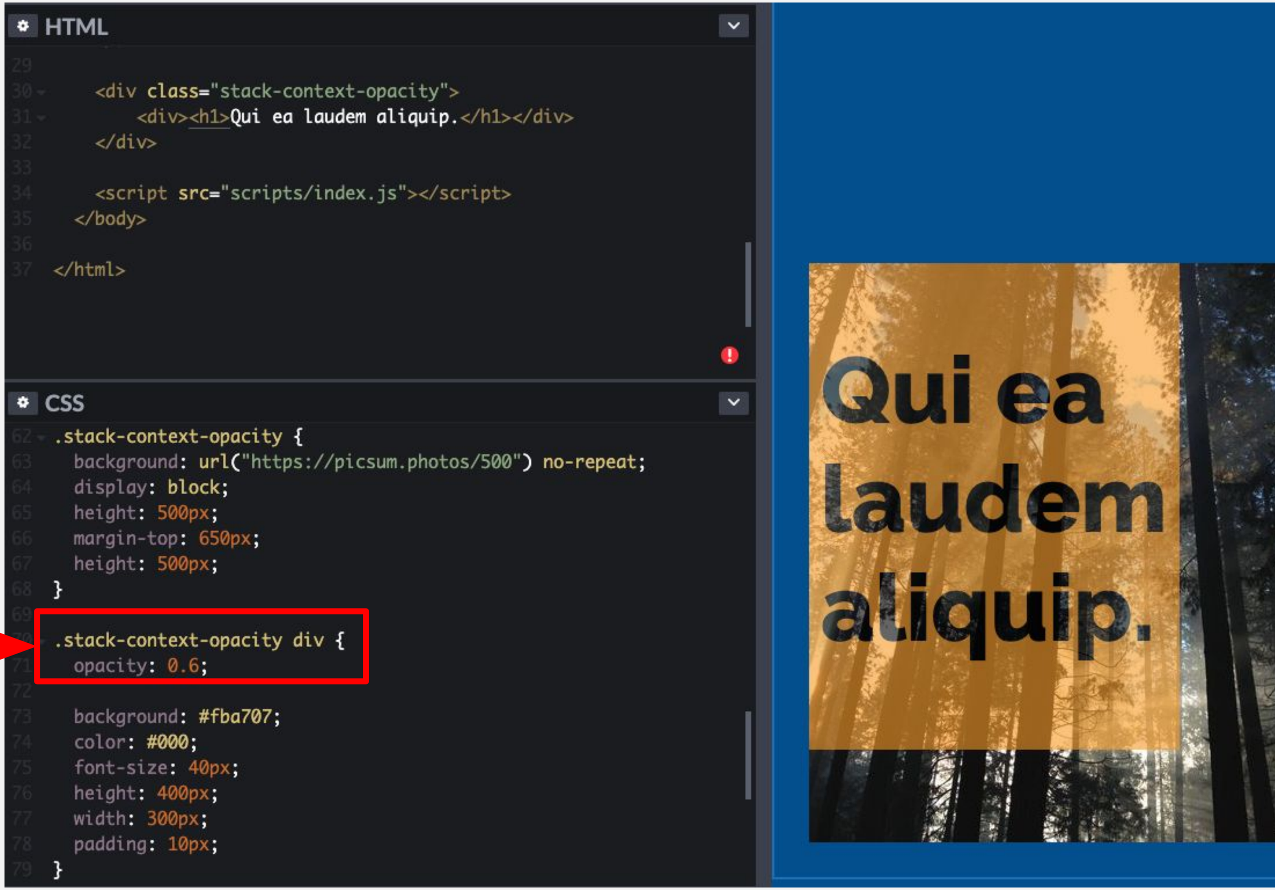
border -

Filter Show all

```
▸ display
  block
```

Stacking context when:

Element with a opacity value less than 1.



```html
    <div class="stack-context-opacity">
        <div><h1>Qui ea laudem aliquip.</h1></div>
    </div>

    <script src="scripts/index.js"></script>
  </body>

</html>
```

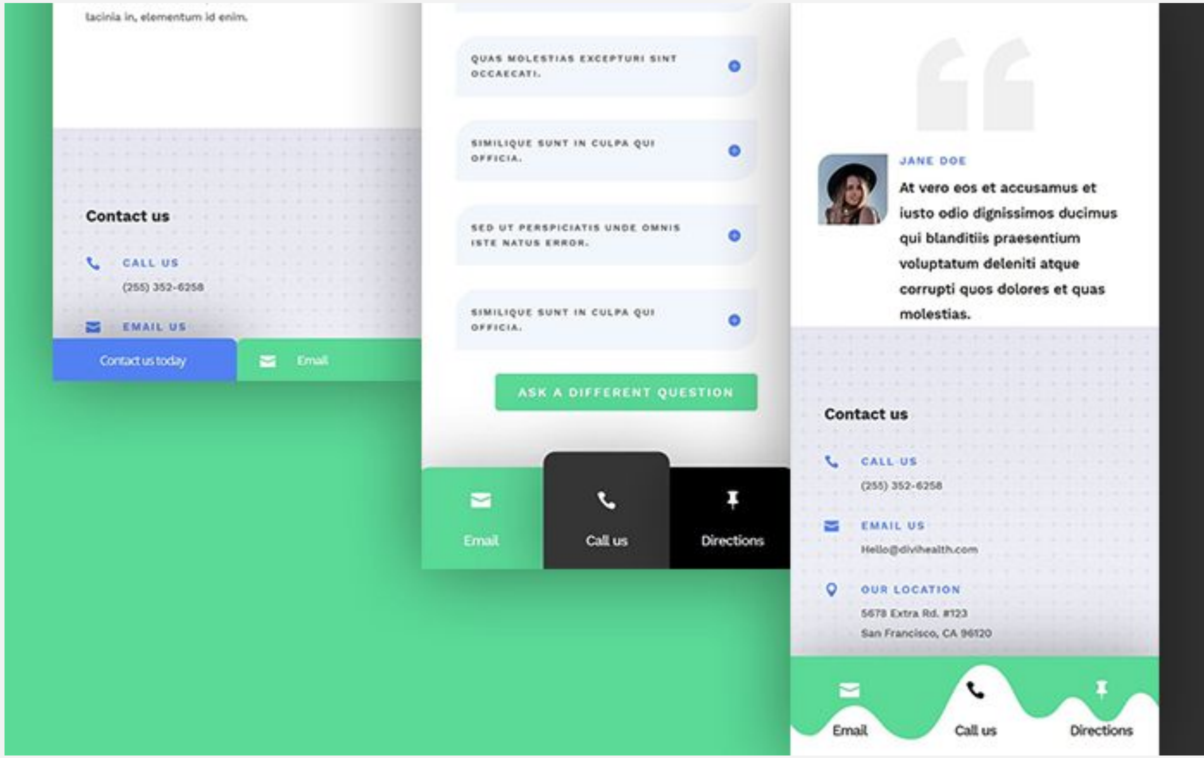
```css
.stack-context-opacity {
  background: url("https://picsum.photos/500") no-repeat;
  display: block;
  height: 500px;
  margin-top: 650px;
  height: 500px;
}

.stack-context-opacity div {
  opacity: 0.6;

  background: #fba707;
  color: #000;
  font-size: 40px;
  height: 400px;
  width: 300px;
  padding: 10px;
}
```

Stacking context when:

Element with a position value fixed or sticky.



At vero eos et accusamus et iusto odio dignissimos ducimus qui blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias.

Stacking context when:

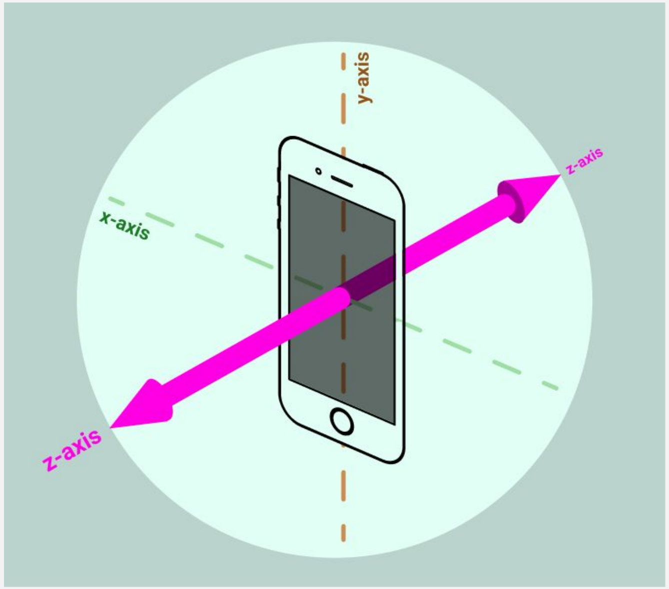
Element with a **position** value absolute or relative and z-index value other than auto.

Z-index

Why so hated?

z-index: -999999;

Z-index



More controlled way to create layers and a visual hierarchy in a 3rd. plane: z-axis.

Z-index

Restrictions:

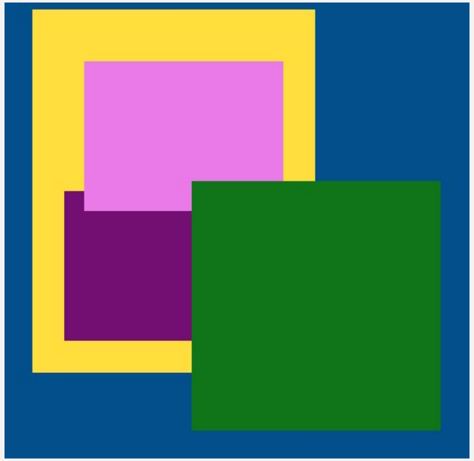
Only works on “positioned elements”.

```
div {
    position: static | relative | absolute | sticky | fixed;
    z-index: 1;
}
```

Z-index

Restrictions:

Z-index only competes with siblings HTML elements.



Here, yellow and green are 2 different stacking context. They are siblings, and no matter how many children the yellow can have, they will always remain below green due to their z-index value.

Let’s code!
codepen.io/mariapazz

Z-index & Positioned Elements

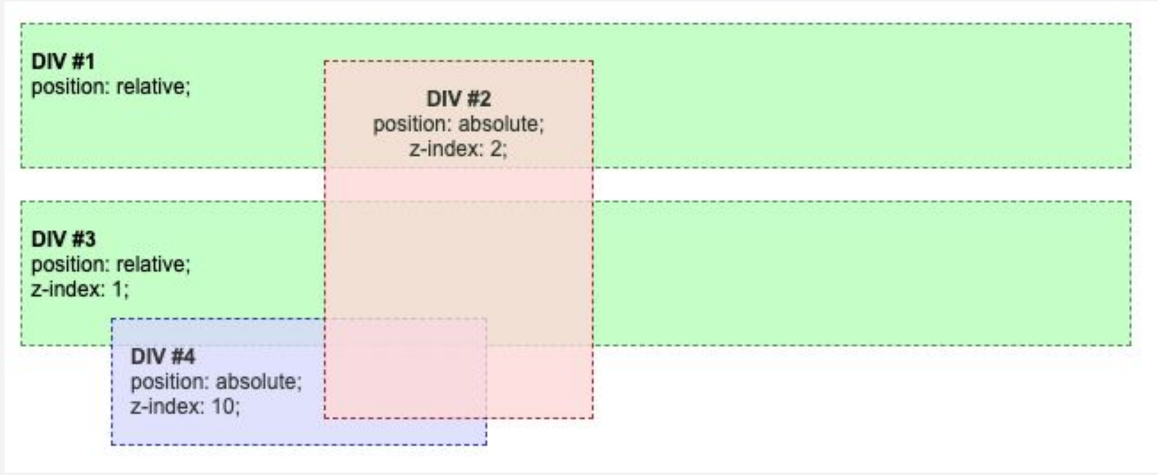
It's over 9.000

Z-index & Positioned Elements

DIV #2 (z-index: 2) is above DIV #3 (z-index: 1), because they both belong to the same stacking context (the root one), so z-index values rule how elements are stacked.

What can be considered strange is that DIV #2 (z-index: 2) is above DIV #4 (z-index: 10), despite their z-index values.

Let’s code! codepen.io/mariapazz



👀 developer.mozilla.org/Stacking_context_example

Z-index & Positioned Elements

Example: how to use ranges for z-index values

- **auto**: Body content.
- **1-49**: Specialized content (ex, super-menu, drop-down menu).
- **50-99**: Fixed positioned elements (ex, header, footer, drop-target).
- **100**: Overlay (ex, fly-out menu).
- **101**: Overlay (ex, modal window system).
- **102**: Overlay (ex, ???).
- **200**: Globally positioned pop-up menu.
- **201**: Globally positioned tool-tip.

👀 www.bennadel.com/1-stacking-context-is-the-key

Conclusions

Oh my brain.

Conclusions

- Think about layers, having in mind when a new Stacking Context is created.
- Root element of DOM creates a stacking context.
- Don’t use random numbers for z-index
- Think about the parent stacking context: over which context my element is being placed.

“And just remember,

if you're tempted to throw in a z-index of 999999, stop, take a step back, and think about why a stacking context would require such a value.”

👀 www.bennadel.com/1-stacking-context-is-the-key

Thanks!

Linkedin // Instagram

@mariapazmp

Brainstorming topics.

Additional documentation and topics:

- Problems that can be solved by understanding the SC concept (z-index: auto)
- Children who are stocked in the context.
- How stacking context in z-index works: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Positioning/Understanding_z_index/Stacking_context_example_2
- Excersie: Create a multi-level menu: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Positioning/Understanding_z_index/Stacking_context_example_3

https://tiffanybbrown.com/2015/09/css-stacking-contexts-wtf/index.html

https://www.bennadel.com/blog/3371-stacking-context-is-the-key-to-understanding-the-css-z-index.htm

https://philipwalton.com/articles/what-no-one-told-you-about-z-index/